Rapport IDM individuel

Thomas Guzik

Ce projet est un fork du projet original
https://gitlab.istic.univ-rennes1.fr/17000402/idm-project

Test

Lancer les tests

Sur Docker :

```
docker build -t idm_test -f Dockerfile.test .
docker run -t idm_test
```

Sans Docker

```
gradle test
```

Commentaires

Comme observé sur la partie commune, nos compilateurs et notre interpréteur ne génèrent pas le même résultat suivant les programmes en entrée.

On distingue 2 types d'erreurs :

- les erreurs d'affichage (par exemple, un compilateur va générer un `\t` alors qu'un autre compilateur va générer un espace)
- les erreurs de fonctionnalités (par exemple bash n'affiche pas le bon résultat lorsque on lui demande de supprimer toutes les lignes et colonnes)

Benchmark

Le benchmark a été fait sur un fichier de 260 colonnes et de 10000 lignes.
Pour chaque temps mesuré, nous mesurons la moyenne du temps de compilation sur 4 fois, et la moyenne du temps d'execution sur 2 fois.

Les fonctionnalités testés par le benchmark sont :

- la suppression
- l'insertion
- la modification
- les fonctions de somme

Lancer les benchmark

Sur Docker :

```
docker build -t idm_becnh -f Dockerfile.bench .
docker run -t idm_bench
```

Sans docker :

```
cd qsv-code
python3 generate.py
cd ..
gradle shadowJar
python benchmark.py
```

Résultat

(La présentation des données a été légèrement changé pour l'adapté au rapport)

	Programme	Langage	Temps
Exec	delete_col	bash	40.340090530498856
Exec	delete_col	python3	20.59580906924748
Exec	delete_col	**int**	14.881956353832114
Exec	delete_line	bash	27.334502474503097
Exec	delete_line	python3	14.089602334752271
Exec	delete_line	**int**	10.290204740002082
Exec	delete_cond	bash	25.91027949400086
Exec	delete_cond	python3	13.345220581251851
Exec	delete_cond	**int**	9.741428278000967
Exec	insert_line	bash	0.8233550179975282
Exec	insert_line	python3	1.182366438997633
Exec	insert_line	**int**	1.9254968309978722
Exec	insert_col	bash	9.901011491503596
Exec	insert_col	python3	5.496361827503279
Exec	insert_col	**int**	5.716849608168559
Exec	update	bash	0.004228980498737656
Exec	update	python3	0.022872893749081413
Exec	update	**int**	0.6805213896671679
Exec	update_cond	bash	0.0023148030013544485
Exec	update_cond	python3	0.01772620950214332
Exec	update_cond	**int**	0.6764605166681577
Exec	compute_col_nb	bash	1.196831569501228
Exec	compute_col_nb	python3	0.9937351400003536
Exec	compute_col_nb	**int**	1.5709642513344686
Exec	compute_col_str	bash	1.028538391001348
Exec	compute_col_str	python3	0.9734893360000569
Exec	compute_col_str	**int**	1.4764686078318239
Exec	compute_line_nb	bash	0.8657249354982923
Exec	compute_line_nb	python3	0.8423104939975019
Exec	compute_line_nb	**int**	1.4600859878313106
Exec	compute_line_str	bash	0.5383259650006949
Exec	compute_line_str	python3	0.7094076554985804
Exec	compute_line_str	**int**	1.3368366129980132

Pour les différentes opérations de suppressions, nous pouvons classer (du plus rapide au plus lent):

1. Interpréteur
2. Python
3. Bash

Pour les différentes opérations de insertions, nous pouvons classer (du plus rapide au plus lent):

1. Python
2. Interpréteur
3. Bash

Pour les différentes opérations de modification, nous pouvons classer (du plus rapide au plus lent):

1. Bash
2. Python
3. Interpréteur

Pour les différentes opérations de somme, nous pouvons classer (du plus rapide au plus lent):

1. Python
2. Bash
3. Interpréteur

Dans l'ensemble Python est le plus rapide, suivi de l'interpréteur et de bash

A présent regardons les temps de compilation :

Compil	delete_col	**sh**	1.8056793164996634
Compil	delete_col	**py**	2.1126152177494077
Compil	delete_line	**sh**	1.869717583002057
Compil	delete_line	**py**	1.8815699140013749
Compil	delete_cond	**sh**	2.001338349749858
Compil	delete_cond	**py**	1.8043725150018872
Compil	insert_line	**sh**	2.6928905992517684
Compil	insert_line	**py**	2.842692013249689
Compil	insert_col	**sh**	6.317588854750284
Compil	insert_col	**py**	11.347161024998059
Compil	**update**	**sh**	1.9711819257518073
Compil	**update**	**py**	2.0504504984983214
Compil	update_cond	**sh**	1.9583979587478098
Compil	update_cond	**py**	1.861129869499564
Compil	compute_col_nb	**sh**	2.036928312250893
Compil	compute_col_nb	**py**	2.0802484639989416
Compil	compute_col_str	**sh**	1.8698919332509831
Compil	compute_col_str	**py**	1.806560335749964
Compil	compute_line_nb	**sh**	1.787689823000619
Compil	compute_line_nb	**py**	1.9021969080022245
Compil	compute_line_str	**sh**	1.879066472998602
Compil	compute_line_str	**py**	1.885766261999379

On observe que le compilateur bash compile le plus rapidement